

\* Random Forest  
\* Decision Tree.

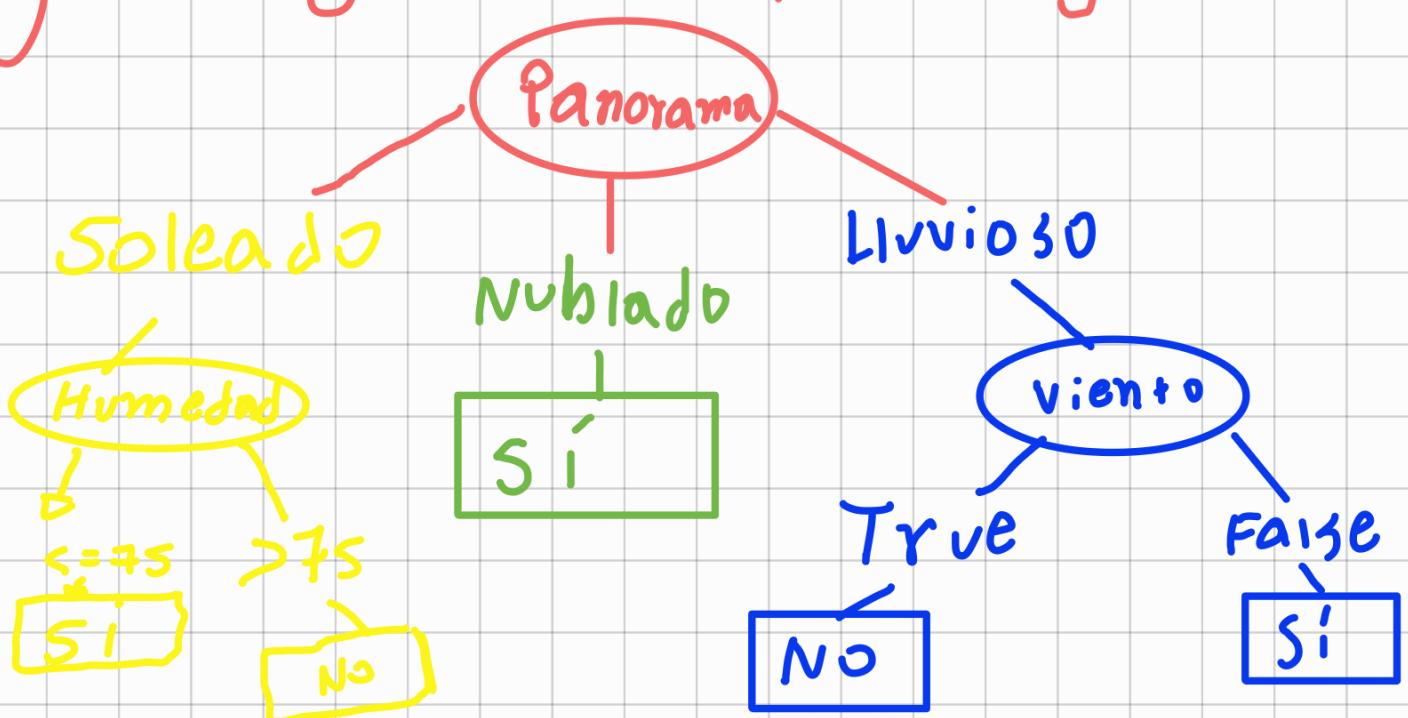
los árboles de decisiones aprenden con Reglas If-else

para los árboles de decisión  
\* Aprendizaje supervisado  
\* Diferentes algoritmos derivan de los Árboles de decisiones

\* Los nodos son:  
son las divisiones

\* Si un nodo no conduce a otro se lo llama nodo terminal o hoja.

Ej: Mejor clima para jugar al tenis:



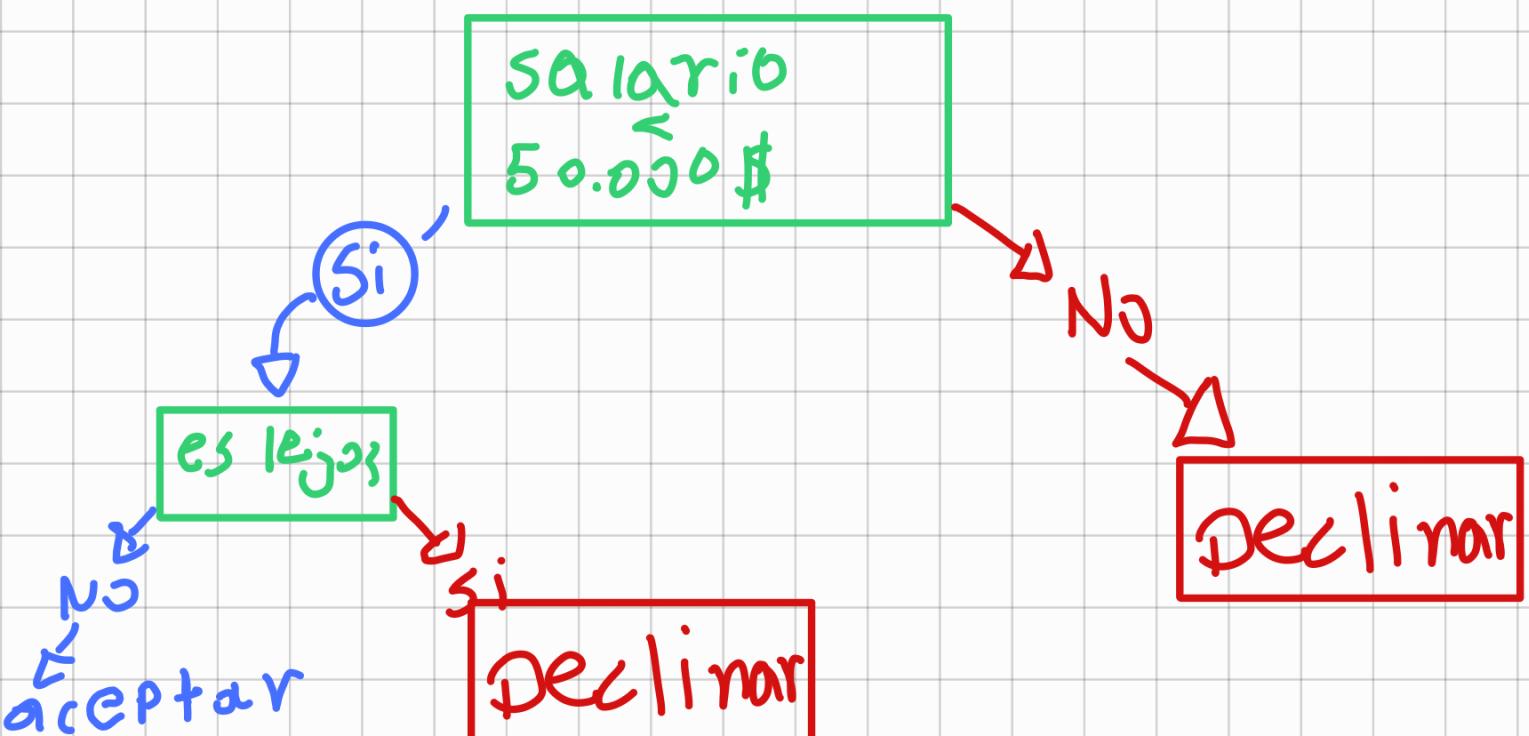
#Creamos un arbol de Decisiones usando python, google colab, Pandas, Matplotlib.

① En la primera parte solo hicimos una pequeña limpieza de datos y tambien una muy breve visualización para luego empezar con SK-learn. separamos los datos en 70%, 30% y empezamos a entrenar los modelos, simplemente importamos nuestra función, le pasamos nuestros datos y calculamos el accuracy

---

Ejemplo de un arbol de Decision

¿Deberia aceptar una nueva propuesta laboral?



las Variables son analizadas.

## Terminología:

- ⊗ Nodo Raíz: la Pregunta main.
- ⊗ División: las dos condiciones Tras el if-else
- ⊗ Nodo de decisión: las siguientes preguntas tras luego de la main
- ⊗ Nodos hoja: Si ya no podemos seguir analizando llegamos a un nodo hoja. e.g: si el trabajo no cumple el salario esperado lo descartamos.
- ⊗ Podar/Pruning: limitamos el algoritmo para que la complejidad no se extienda demasiado.
- ⊗ Nodo Padre/hijo/madre.

---

¿Cuando Usamos Decision Tree?

- ⊗ Algoritmo fácil de entender (whitebox)
- ⊗ Resultados fáciles y rápidos de entender.
- ⊗ Da paso a otro algoritmo como Random Forest

## Desventajas de DT:

- 1-) Si tenemos pocos datos es peligroso trabajar con DT. porque tiende a over fitting.
- 2-) Se ve influenciado por valores atípicos (outliers) Importantísimo
- 3-) Árboles muy complejos llevan a un mal resultado.
- 4-) Sesgos con datasets no bien balanceados.

---

A pesar de las desventajas:

- Sencillo de entender
- Funciona bien con una gran cantidad de datos
- Robusto.
- Un método muy útil para datos cuantitativos.
- API. CO A Clasificación / Regresión.

# • Sería útil aprender sobre XGBoost

# A esta altura de el curso empezamos con el proyecto final: un algoritmo de clasificación de D.T.

Así que apartir de ahora estare anotando cosas relevantes del código

1-) Importamos las librerias

2-) Analizamos nuestros datos:

Nota: la función **Value\_counts** de pandas es muy útil. También la función **is-null().sum()** para ver los valores missing.

3-) Separamos nuestros datos en 70/30

3.1-) como buena práctica volvemos a analizar los datos.

4-) Convertimos nuestros datos tipo "Object" a tipo **int64** usando una libreria llamada **category-encoders**.

Para hacer eso hacemos lo siguiente:

```
import category_encoders as ce
```

```
encoder = ce.OrdinalEncoder(cols=[columns])
```

```
XTrain = encoder.fit_transform(XTrain)
```

```
XTest = encoder.fit_transform(XTest)
```

}

5-) Importamos la librería para entrenar nuestro modelo de árbol de decisión

5.1-) Usamos D.T (después usaremos RDF)

6-) Entrenamos el modelo de D.T. y calculamos las predicciones para Train y también para test.

Aquí abrimos un pequeño parentesis para hablar de ¿cómo evaluamos un modelo de árbol de decisión?

- Error
- Correcto

### • Matriz de Confusión:

1) Esto permite ver el desempeño del modelo.

2.) Representa las predicciones de cada clase.

Valores Prediccion	Valores Reales	
	Falsos Positivos	Falsos Negativos
Verdaderos Positivos	Verdaderos Negativos	Negativos

Formula para el accuracy

$$\frac{VP + VN}{(VP + FP + FN + NN)} * 100\%$$

Acc bueno

80) ~ 90%

Formula de precisión

$$\frac{VP}{VP + FP}$$

O Recall

Sensibilidad : Tasa de verdaderos positivos . Proporción de casos Positivos que fueron correctamente identificados

---

$$\frac{VP}{VP + FN}$$

Especificidad : Tasa de verdaderos Negativos . Proporción de casos negativos que fueron correctamente identificados

---

$$\frac{VN}{Vr + FP}$$

F1-Score: Resumen la presión y la sensibilidad en una sola métrica

$$2: \frac{\text{Precision} \cdot \text{recall}}{\text{Precision} + \text{recall}}$$

Como podemos ver todas estas métricas derivan de la matriz de confusión.

# Random Forrest:

Algoritmo de ensamble, Combinamos  
multiples arboles de decisiones.